(4)

ISI Research Report
ISI/RR-88-213
April 1989

University of Southern California

John Yen
Robert Neches
Robert MacGregor

Classification-based Programming: A Deep Integration of Frames and Rules

INFORMATION SCIENCES INSTITUTE
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT This document is approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ISI/RR-88-213	5. MONITORING ORGANIZATION REPORT NUMBER(S) -------
6a. NAME OF PERFORMING ORGANIZATION USC/Information Sciences Institute; 6b OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION -------
6c. ADDRESS *(City, State, and ZIP Code)* 4676 Admiralty Way Marina del Rey, CA 90292-6695	7b. ADDRESS *(City, State, and ZIP Code)* -------
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA Air Force Logistics Command; 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-86-C-0178 F33600-87-C-7047
8c. ADDRESS *(City, State, and ZIP Code)* (OVER)	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
-------	-------	-------	-------

11. TITLE *(Include Security Classification)*
Classification-based Programming: A Deep Integration of Frames and Rules (Unclassified)

12. PERSONAL AUTHOR(S) Yen, John; Neches, Robert; MacGregor, Robert

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Research Report	FROM ______ TO ______	1989, April	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
09	02	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
classification-based programming, control knowledge, hybrid architecture, knowledge representations, rule-based paradigm

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Rules and frames are two knowledge representation schemes whose strengths and weaknesses are complementary to each other. Although several systems have attempted to integrate the two, few efforts have been made to incorporate the classification reasoning of the frame representations into the rule-based systems. To achieve a deep integration of the two schemes, we have developed and implemented a *classification-based programming* paradigm where the rules' LHS and the functionalities of their RHS are represented in the terminological space. The major processes of the architecture consist of a pattern matcher that is driven by semantic representations rather than by structural patterns, a rule base organizer that classifies rules and rule classes into a taxonomy, a conflict set manager that filters rules, and a rule interpreter that selects and executes rules. The architecture facilitates the representation of control knowledge by inferring the specificity of rules and by inheriting the control strategies from rule classes. The paradigm not only enhances the reasoning capabilities of rule-based systems, but also encourages explicit representation of various kinds of knowledge implicit in rules so that they can be shared by different domains and used for multiple purposes.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Victor Brown Sheila Coyazo	22b. TELEPHONE *(Include Area Code)* 213/822-1511; 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

8C.

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Air Force Logistics Command
Wright-Patterson Air Force Base
Ohio 45433-5320

Accession For	
NTIS GRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

ISI Research Report
ISI/RR-88-213
April 1989

University of Southern California

John Yen
Robert Neches
Robert MacGregor

Classification-based Programming: A Deep Integration of Frames and Rules

INFORMATION SCIENCES INSTITUTE

213/822-1511

4676 Admiralty Way/Marina del Rey/California 90292-6695

This research was supported by the Defense Advanced Research Projects Agency under Contract No. MDA903-86-C-0178 and by the Air Force Logistics Command under Contract No. F33600-87-C-7047. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, the Air Force Logistics Command, the U.S. Government, or any person or agency connected with them.

1. Introduction

Rules and frames are two contrasting schemes for representing different kinds of knowledge. Rules are appropriate for representing logical implications, or for associating actions with conditions under which the action should be taken. In this paper, we focus our discussion on the second kind of rules (i.e., action rules). Frames (or semantic nets) are appropriate for defining terms and for describing objects and the taxonomic class/membership relationships among them. An important reasoning capability of frame systems with well-defined semantics is that they can infer the class/membership relationships between frames based on their definitions [Brachman 85, Moser 83].

Since the strengths and weaknesses of rules and frames are complementary to each other, a system that integrates the two will benefit from the advantages of both techniques. This paper describes a hybrid architecture called *classification-based programming*, which extends the production system architecture using automatic classification capabilities within frame representations. In doing so, the system enhances the power of a pattern matcher in a production system from *symbolic matching* to *semantic matching*, organizes rules into rule classes based on their functionalities, and infers the various relationships among rules that facilitates explicit representation of the control knowledge.

Section 2 briefly reviews the existing AI programming architectures related to our work. Section 3 describes the architecture of our classification-based programming paradigm. Section 4 describes our implementation of the architecture and a small expert system built utilizing it. Section 5 discusses related research issues and our plans for addressing them.

2. Problems with existing AI programming architectures

Rule-based systems have encountered a number of criticisms. First, rules are often used to implicitly represent contexts, control knowledge, and structural knowledge [Clancey 83, Aikins 80]. Second, the meaning of the terminology used by the rules is often ill-defined [Swartout and Neches 86], Third, it is difficult to structure large rule sets [Fikes and Kehler 85]. These problems make rule-based systems limited in explaining their reasoning and costly to maintain [Swartout 83, Neches, et al. 85].

A major distinction among the frame-based systems is the *semantics* associated with the slots. KRL [Bobrow 77], FRL [Roberts 77], and FRAIL [Charniak 83] are early frame-based systems that have been criticized for lacking well-defined semantics [Woods 75, Brachman 83]. As a result, KL-ONE [Brachman 85] and and its descendant NIKL [Moser 83] have well-defined semantics for *concepts* -- frames that represent unary predicates (i.e., one-place predicates). We will refer to the earlier frame systems that represent only primitive concepts as *weak semantic frame-based systems*, and the latter ones that supports both primitive and defined concepts as *strong semantic frame-based systems*.

The major advantage of a strong semantic frame-based system is its special-purpose reasoner: the *classifier* [Schmolze and Lipkis 83]. The classifier positions a defined concept in the subsumption lattice such that (i) it is below all concepts that subsume it, and (ii) it is above all the concepts that it subsumes. Thus, the taxonomies of strong-semantic frame-based systems are inferred by the system, not specified by users as with weak-semantic frame-based systems.

Although frame-based systems provide a convenient way to organize data objects in a class taxonomy and to inherit their properties from more general classes, they are weak in representing and using rule knowledge. Early frame-based systems used procedural attachment (i.e., demons) to deal with limited classes of rules, but these do not provide a general control structure.

KEE, ART, and Knowledge Craft are hybrid systems that combine weak-semantic frame-based knowledge representation with a rule language. These systems demonstrated some advantages of integrating the frame representation and the rule language [Fikes and Kehler 85]. The frames provide a rich structural language for describing the objects referenced in the rules. Frame taxonomies can be used to partition, index, and organize a system's production rules. Since these systems do not distinguish between primitive concepts and defined concepts, they can not further integrate the automatic classification components of strong-semantic frame representation with rule-based systems.

KRYPTON [Brachman, Fikes, and Levesque 83], KL-TWO [Vilain 84], and BACK [Von Luck 85] are three strong-semantic hybrid architectures. All of them have two components for representing two different kinds of knowledge: *terminological* and *assertional*. Because strong-semantic frame languages are appropriate for representing and reasoning about term definitions, they serve as the terminological language of all three systems. They differ, however, in their choices of an assertional language.

The *realizer* is an important special-purpose reasoner that both KL-TWO and BACK use for "gluing" assertions and term definitions together. The *most specific generalization* (MSG) of an instance is the intersection of all concepts that it belongs to. By computing and recording the MSG of an instance, the system can quickly identify all the terminological concepts that describe an instance. No effort, however, has been made to integrate the KL-TWO or BACK realizers with the pattern matcher of rule-based systems.

KL-TWO provides a *noticer* mechanism for users to define demons that get executed when the conditions are met [Vilain 84]. Even though the noticer has improved the expressive power and control of demons, it lacks a global control mechanism like the recognize-act cycle of production systems. Moreover, the control knowledge is represented in ways that are difficult to share, explain, and reason about.

3. The classification-based programming architecture

The goal of the *classification-based programming* paradigm is to represent rule knowledge within the framework of strong-semantic frame representations. By doing so, we can use the automatic classification component of the frame system to extend the rule system's capability in pattern matching, in organizing rules, in controlling rules, and in resolving conflicts.

3.1. The CONSUL Mapper

The work done in the CONSUL system [Mark 80, Mark 81] is the first attempt to integrate rules into strong-semantic frame-based systems. The major contribution of CONSUL's rule-based inference, which is built in NIKL, are the following:

1. It uses the classifer to match data with the rules' LHS.
2. It uses the taxonomic structure of the knowledge base to infer the specificity among rules.
3. It organizes and selects rules based on the structure of the knowledge base.[2]

The function of CONSUL rules is strictly to create new descriptions that redescribe a user request until the system can understand it and respond to it. Although the paradigm can be used for any applications, its inference architecture and its RHS actions are not as general as that of a production system. For example, the conflict set is always the set of rules that can redesicribe a given description. Because of this, CONSUL does not need to employ the conflict resolution strategies of production system for rule selection. Moreover, because of the limitation of the underlying knowledge representation system (i.e., NIKL), CONSUL does not distinguish terminological knowlege (e.g., terms referred by the rules' LHS and RHS) from assertional knowledge (e.g., input data that match the rules' LHS and the descriptions created by the rules' RHS).

3.2. An architecture of the classification-based production system

Figure 3-1 shows the general architecture of our classification-based production system. Rules and facts are stored in a rule base and a facts database, respectively. The architecture has four major processes: the rule base organizer, the facts manager, the conflict set manager, and the rule interpreter. The rule base organizer translates the user's definitions about rule classes and individual rules into their internal representations and organizes them into a rule taxonomy in the rule base. The facts manager updates the facts database whenever the factual knowledge is modified. An important component of the facts manager is a semantic pattern matcher that detects changes to the conflict set that arises from the changes to the assertional data. Based on these conflict set changes, the conflict set manager constantly updates the conflict set. The rule interpreter selects productions from the conflict set and executes their

[2]This part of CONSUL work has not been published or fully implemented.

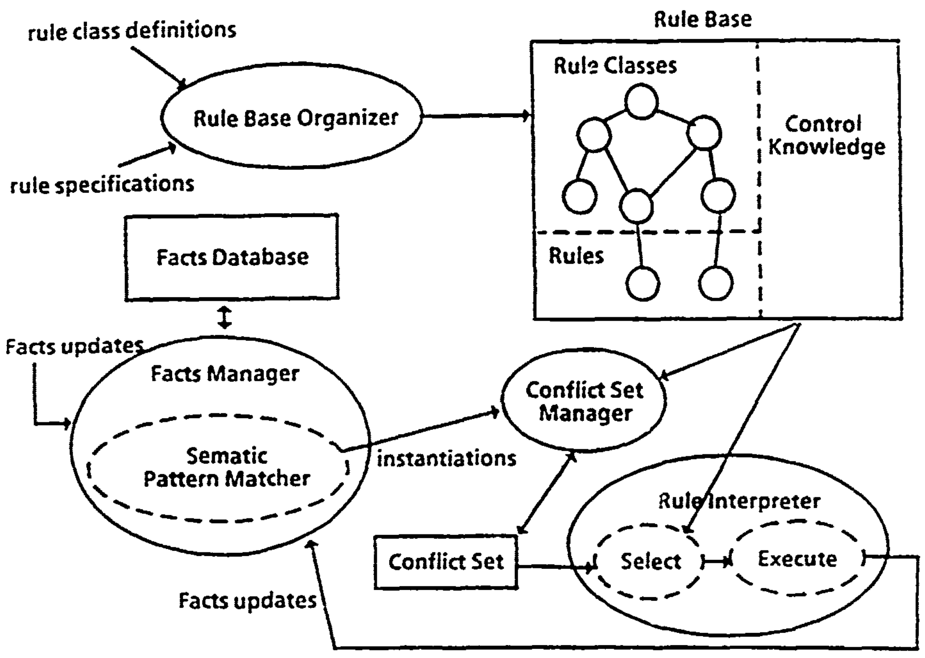

Figure 3-1: The architecture of the classification-based production system

right-hand side actions. Two kinds of control knowledge are stored in the rule base: one for the conflict set manager to filter instantiations, and one for the rule interpreter to select instantiations.

Like the conventional production system, the select and execute operations of the rule interpreter repeat until the conflict set is empty or a **stop** command is executed. The right-hand side of a production often modifies the facts database. Therefore, executing a production usually triggers the semantic pattern matcher, which in turn causes the conflict set to be updated. As we will see in the following discussion, the architecture extends the conventional rule-based paradigm through the use of the classification capabilities of the strong semantic frame-based systems.

3.3. Representing productions in frames

Before we can take advantage of the inferential power of the classifier and the realizer in rule-based systems, we need to represent knowledge about rules in a fashion so that the frame system can reason about it. A production rule has two major components: a left-hand side (LHS) that describes its triggering condition, and a right-hand side (RHS) that contains the actions to be executed when the triggering condition is met. In the classification-based production system, the LHS of a production is constructed using the terminological language, and the RHS has a dual representation: a declarative one that explicitly captures the RHS actions in frame representation and a procedural one that can be executed by the rule interpreter.

3.3.1. The Left-hand Side

The left-hand sides (LHS's) of productions are constructed using terminological concepts. The function of the rule's LHS condition and the terminological concepts are similar because both of them specify a set of data. Therefore, we can use concepts in the terminological space to construct the LHS of a rule. The simplest way to do this is to translate the LHS into a concept definition, then define a rule whose LHS slot simply points to the concept. For example, the LHS condition of rule R1 shown in Figure 3-3 "X is a discrepancy whose status is inactive" can be represented as the following NIKL concept definition:

```
(defconcept inactive-discrepancy (Specializes discrepancy)
        (restriction status (VR inactive)) ).
```

Often the LHS of a rule describes certain complex relationships among several different kinds of data objects. In the conventional production system, this complex condition is represented by several *condition elements*. In the classification-based production system, similar complex conditions can be represented by a concept whose meaning is specified in terms of its relationships to other concepts. For example, the LHS of a rule in [Brownston et. al 85, p. 122] can be stated in English:

```
IF    there is an active goal to be on an object, o1
  and the object o1 is at location p
  and the monkey is at location p holding some object, not nil
THEN ...
```

An OPS5 representation and a NIKL representation of the LHS are shown in Figure 3-2. The NIKL concept not-to-hold-goal represents the above LHS condition. Although the NIKL representation seems more complex than the OPS5 approach, the former captures more knowledge than the latter does (e.g., the fact that the **At** attribute of **Phys-obj** and **Monkey** are inherited from **Real-world-object** is not modeled in OPS5 LHS).

Since OPS5 and NIKL can be viewed, respectively, as typical programming languages for conventional rule-based systems and for classification-based systems, this example illustrates several differences between the conventional representation of the LHS and the classification-based representation:

1. Rather than capturing the relationships between objects by sharing variables across several condition elements, the relationships are explicitly captured using *role restrictions* (e.g., the **Subject** of **Not-to-hold-goal** is a **Full-handed-monkey**) and *constraints* (e.g., the **Subject** and the **Object** of the goal are **At** the same **Location**) in the classification-based approach.

2. The conventional LHS could have several free variables, while the LHS in our approach contains only one free variable and, hence, is always centered around one class of objects.

3. The classification-based approach reduces maintenance cost and facilitates explanation by sharing knowledge. For instance, the **Full-handed-monkey**

OPS5:

```
(goal        ↑status active ↑type on   ↑object-name <o1>)
(phys-object ↑name   <o1>    ↑at   <p>  )
(monkey      ↑at     <p>    ↑holds <> nil)
```

NIKL:

```
(DEFCONCEPT real-world-object PRIMITIVE)
(DEFROLE at (DOMAIN real-world-object) (RANGE  location))

(DEFCONCEPT phys-object PRIMITIVE (SPECIALIZES real-world-object)

(DEFCONCEPT animal PRIMITIVE (SPECIALIZES real-world-object))
(DEFROLE hold-obj (DOMAIN animal) (RANGE real-world-object))
(DEFCONCEPT monkey PRIMITIVE (SPECIALIZES animal))
(DEFCONCEPT full-handed-monkey (SPECIALIZES monkey)
    (restriction hold-obj (MIN 1)) )

(DEFCONCEPT goal PRIMITIVE)
(DEFROLE status  (DOMAIN goal) (RANGE  goal-status))
(DEFROLE type    (DOMAIN goal) (RANGE  goal-type))
(DEFROLE object  (DOMAIN goal) (RANGE  real-world-object))
(DEFROLE subject (DOMAIN goal) (RANGE  animal))

(DEFCONCEPT not-to-hold-goal (SPECIALIZES goal)
    (RESTRICTION status  (VR active))
    (RESTRICTION type    (VR on))
    (RESTRICTION subject (VR full-handed-monkey))
    (= (object at) (subject at))
```

Figure 3-2: The OPS5 anu NIKL representation of a LHS

concept in Figure 3-2 can be reused in other LHS conditions referencing the monkey holding something.

4. Patterns are classified into a subsumption lattice in a classification-based system. As we will see below, this enables classifying rules, performing semantic matching, determining "more specific than" relationships among rules, and performing conflict resolution in the recognize-act cycle.

3.4. The rule base organizer

Most rule systems either provide no mechanism for organizing rules or ask the user to specify tl rule classes each rule belongs to. The major functions of our rule base organizer are to construct a taxonomy of rule classes and to group the rules into those classes for the user. It does so by representing the knowledge for organizing rules explicitly, and using the classification capabilities of the strong-semantic frame systems to classify rules and rule classes. Therefore, the rule base organizer encourages explicit representation of a rule set's organization.

Rules can be grouped by their triggering conditions, the goal of their actions, their conflict resolution strategies [Neches 87], or other criteria. In order to classify rules into rule classes, the system must have appropriate slots that relate the rules and rule classes to the classification criteria, and an appropriate knowledge base that captures the meaning and the abstraction of the fillers to the slots. For example, a rule class containing all rules that are triggered by goal states in a search problem can be defined by the following NIKL concept definition:

```
(DEFCONCEPT Detect-goal-state-rule-class (SPECIALIZES Production)
    (RESTRICTION LHS (VR Goal-state)))
```

where the role LHS relates a production to its left-hand side.

The maintenance of a large unorganized rule base is costly because it is difficult to locate a particular rule, to find rules whose tasks are similar or related to a given rule, or to activate only a subset of rules for testing or debugging. Thus, organizing rules in a way meaningful to the rule base developer helps in retrieving, browsing, and testing the rule base. With the rule organization, control knowledge needs to be described only at the abstract level of rule classes. Each rule automatically inherits the control knowledge from all its classes. Moreover, because the rule classes themselves are organized into a superclass/subclass lattice, the control knowledge of a more specific class could override that of a general class. We will return to this point in section 3.6.

3.5. The semantic pattern matcher

Representing the LHS by using terminological concepts enables the production system to extend its pattern matching capabilities from *symbolic matching* to *semantic matching* by using the realizer. Semantic matching extends symbolic matching in that it utilizes the terminological knowledge for pattern matching. Although the terminological knowledge can be represented as a large set of rules that infer the membership relationships among classes (i.e., terms), representing them declaratively has two major advantages. First, it is computationally more efficient than triggering a long chain of rules, because most of the subsumption relationship has been precomputed by the classifier. Second, it is easier to modify, maintain, and understand the knowledge because the sufficient and the necessary conditions of a class are represented as a unit rather than as a set of independent rules.

An example of the semantic pattern matcher is matching class variables (as opposed to pattern variables). Early languages (e.g., PSG [Newell 75]) had class variables; later languages moved away from this for efficiency reasons. A classifier does match efficiently, and also deals with recognizing when an item is a class member even when it has not been explicitly declared as such.

To illustrate the advantage of semantic pattern matcher, let's consider rule R1 in Figure 3-3. Suppose we have an expert system that detects problematic situations (called discrepancies) and attaches notes to them. A discrepancy could be in exactly one of the four states: **Unsolved, Pending, Explained,** or **Solved.** These states are further grouped into two categories. An **Active** discrepancy is either **Unsolved** or

```
R1: IF X is a DISCREPANCY whose status is INACTIVE,
    THEN make the status of the NOTE attached to X INACTIVE.

R2: IF X is a DISCREPANCY whose status is SOLVED,
    THEN remove the NOTE attached to X.
```

Figure 3-3: The example of a specific rule overrides a general rule

Pending. An **Inactive** discrepancy is either **Explained** or **Solved.**

Figure 3-3 shows two rules in the system that manages the notes when the status of their associated discrepancies have been changed. For readability, frames are printed in bold uppercase and slot names are printed in italic lowercase.

Assume that the status of a discrepancy has been changed to **Explained.** A symbolic pattern matcher would not find a match with the LHS of R1, because the symbol **Explained** does not correspond to the symbol **Inactive.** However, a pattern matcher utilizing a realizer would find a match with R1's LHS because the terminological knowledge (i.e., that the **Explained** status is a kind of **Inactive** status) would allow it to infer that an explained discrepancy is also an inactive discrepancy. This example shows that the inferential power of the conventional pattern matcher can be greatly enhanced by the realizer in a classification-based production system.

3.6. Representing the control knowledge

The architecture facilitates the representation of control knowledge in two ways: (1) control knowledge can be described at the abstract level of rule classes, and (2) the architecture infers the specificity of rules. Control knowledge includes strategies for filtering certain instantiations from the conflict set, and the conflict resolution strategy used by the rule interpreter.

Describing control strategy at the rule class enables us to explicitly state the criteria for using the strategy. Conflict resolution strategy is one kind of control knowledge that has been associated with rule classes [Neches 87]. Our architecture extends Neches' approach by inferring the rule classes of each rule, rather than having the user specify them.

Given production rules R1 and R2, we will call rule R1 *more specific than* rule R2 if the set of R1's rule classes properly contains the set of R2's rule classes.[3] Specificity of rules is useful both for conflict resolution and for filtering rules. Specificity is a classic conflict-resolution heuristic used by languages from OPS through

[3]Because the classifier has pre-computed the membership relationships between all rules and rule classes, the "more specific than" relation between two rules can be computed by a simple subset test.

ART for selecting productions [McDermott 78]. Our approach gives specificity a definition based on semantics, where previously it was definable only in terms of structural correlates like number of condition clauses.

A useful control strategy for the conflict set manager is to include in the conflict set **only** the most specific rule in a rule class for each instance (i.e., filter out other general rules). Applications using this strategy can formulate sets of *general-purpose* productions designed to respond to situations using broad-coverage procedures, and can also formulate additional *special-purpose* productions that invoke procedures tailored to particular situations. This control strategy can be implemented in classification-based programming by associating the most-specific-rule strategy with the desirable rule classes.[4] This is illustrated in the following example.

Let's consider the two rules about discrepancy in Figure 3-3. Suppose both R1 and R2 belongs to the class of rules that manage notes about discrepancies. By associating the most-specific-rule strategy with the rule class, we specify that only one rule in the class can be triggered by a discrepancy. Now, suppose a previously active discrepancy D1 is solved. Conventional conflict-resolution algorithms can not detect that the rule R2 is more specific than R1. In classification-based programming, because the conflict set manager could infer that R2 is more specific than R1, it could therefore prefer R2.

4. The implementation of a classification-based production system architecture

Our current implementation of the classification-based production systems architecture is built on top of NIKL [Kaczmarek 86, Robins 86], a KL-ONE-style knowledge representation language. Because NIKL does not have an assertional component, we model instances as primitive concepts subsumed under **Instance.** The NIKL classifier is used both by the rule base organizer and the semantic pattern matcher of the system.

A NIKL concept has three kinds of slots: *Roles*, *Data*, and *Idata*. The roles are KL-ONE style definitional slots. Data and Idata both link concepts to non-definitional properties. Data is local to the concept, while Idata is inherited through the subsumption hierarchy.

4.1. Representing productions

A production rule is represented as a subconcept of the concept **Production** with two roles (i.e., **LHS** and **RHS-task**) and two nondefinitional slots (i.e., **Bindings** and **RHS-body**). The former two slots are used by the rule base organizer to classify rules into the rule taxonomy; the latter two slots are used by the rule interpreter to execute the RHS actions. The **LHS** of a production is a concept that intentionally defines the

[4] [Neches, et al. 85] describe a system which uses a classifier to prefer more-specific over more-general rules. However, that system does not exhibit a production-rule architecture.

Another issue is that the dual representations of the right-hand side are treated independently at present. It is the responsibility of the knowledge base developer to make the two representation consistent with each other. Hence, a facility that aids the user in formulating a declarative RHS that is consistent with the procedural RHS would be very useful.

6. Summary

We have presented the general architecture and a prototype implementation of the classification-based programming paradigm. Our main objective is to extend the benefits of classification capabilities in strong-semantic frame systems to the developers and users of rule-based systems. By representing the left-hand sides and the functionalities of the right-hand sides in the terminological spaces, the paradigm improves conventional rule-based programming in several aspects. The pattern-matching operation is based on the terminological definitions of the symbols, not the symbols themselves. Rules and rule classes are organized according to their declarative representation. Control knowledge that uses the subsumption relationships among rules and rule classes can be specified at the abstract level of rule classes. The paradigm encourages the development of a rich and coherent knowledge base about conditions and actions, which is shared across rules; this can be used for other important tasks, such as knowledge acquisition and explanation generation.

Acknowledgements

We would like to thank Paul Rosenbloom and John Granacki for their comments on earlier drafts of the paper.

References

[Aikins 80] J. S. Aikins, *Prototypes and Production Rules: A Knowledge Representation for Computer Consultations,* Department of Computer Science, Stanford University, Technical Report STAN-CS-80-814 , 1980.

[Bobrow 77] D. G. Bobrow, and T. Winograd , " An overview of KRL, a knowledge representation language ," *Cognitive Science* 1, (1), January 1977, 3-46.

[Brachman 83] R. J. Brachman , " What IS-A is and isn't: An analysis of taxonomic links in semantic networks ," *Computer* 16 , (10), October 1983 , 30-36 .

[Brachman 85] Brachman, R.J., and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science,* August 1985, 171-216.

[Brachman, Fikes, and Levesque 83] Ronald Brachman, Richard Fikes, and Hector Levesque, "KRYPTON: Integrating Terminology and Assertion," in *Proceedings of the AAAI-83,* pp. 31-35, 1983.

[Brownston et. al 85] Brownston, L., R. Farrell, E. Kant, and N. Martin, *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming,* Addison-Wesley, 1985.

[Charniak 83] E. Charniak, M. K. Gavin, and J. A. Hendler, *The FRAIL/NASL reference manual* , Department of Computer Science, Brown University, Technical Report CS-83-06, 1983 .

[Clancey 83] W. Clancey, "The Epistemology of a Rule-Based Expert System: A Framework for Explanation," *Artificial Intelligence* 20 , (3), 1983, 215-251 .

[Fikes and Kehler 85] Fikes, R., T. Kehler, "The Role of Frame-based Representation in Reasoning," *Communication of the ACM,* (9), September 1985.

[Kaczmarek 86] T. Kaczmarek, R. Bates, G. Robins, "Recent Developments in NIKL," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence,* AAAI, Philadelphia, PA, August 1986.

[Mac Gregor and Bates 87] Robert Mac Gregor and Raymond Bates, *The Loom Knowledge Representation Language,* USC/Information Sciences Institute, Technical Report ISI/RS-87-188, 1987.

[MacGregor 88] MacGregor, R. M., "A Deductive Pattern Matcher," in *Proceedings of AAAI-88,* 1988.

[Mark 80] William Mark, "Rule-Based Inference in Large Knowledge Bases," in *Proceedings of the National Conference on Artificial Intelligence,* AAAI, August 1980.

[Mark 81] William Mark, "Representation and Inference in the Consul System," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* pp. 375-381, IJCAI, 1981.

[McDermott 78] McDermott, J., and C. Forgy, *Production System Conflict Resolution Strategies*, Academic Press, New York, , 1978.

[Moser 83] M.G. Moser, "An Overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding*, Bolt, Beranek, and Newman, Inc., Cambridge, MA, 1983. BBN Technical Report 5421.

[Neches 87] Robert Neches, *Learning through Incremental Refinement of Procedures*, The MIT Press, 1987.

[Neches, et al. 85] Neches, R., Swartout, W., and Moore, J., "Enhanced maintenance and explanation of expert systems through explicit models of their development," *Transactions On Software Engineering* SE-11, (11), November 1985, 1337-1351.

[Newell 75] Newell, A., and J. McDermott, *PSG manual*, Department of Computer Science, Carnegie-Mellon University, Technical Report, 1975.

[Roberts 77] R. B. Roberts, and I. P. Goldstein , *The FRL Manual*, MIT Artificial Intelligence Laboratory, Technical Report AIM-408, 1977.

[Robins 86] Gabriel Robins, The NIKL Manual, 1986.

[Schmolze and Lipkis 83] James Schmolze and Thomas Lipkis, "Classification in the KL-ONE Knowledge Representation System," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

[Swartout 83] W. Swartout, "XPLAIN: A System for Creating and Explaining Expert Consulting Systems," *Artificial Intelligence* 21 , (3), September 1983, 285-325 .

[Swartout and Neches 86] William Swartout and Robert Neches, "The Shifting Terminological Space: An Impediment to Evolvability," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, August 1986.

[Vilain 84] Marc Vilain, *KL-TWO, A Hybrid Knowledge Representation System*, Bolt Beranak and Newman, Technical Report 5694, September 1984.

[Von Luck 85] K. v. Luck, B. Nebel, C. Peltason, A. Schmiedel, *The BACK-System*, Technische Universitaet Berlin, Berlin, West Germany, 1985.

[Woods 75] William A. Woods, *Whats's in a Link: Foundations for Semantic Networks*, Academic Press, 1975.